JERBOA : un modeleur géométrique à base de règles

Thomas Bellet, Agnès Arnould, Pascale Le Gall

JERBOA : un modeleur géométrique à base de règles

Thomas Bellet[1], Agnès Arnould[1], Pascale Le Gall[2]
[1]XLIM-SIC, UMR CNRS 6172, Université de Poitiers, France
[2]MAS, École Centrale Paris, France

Mots-clés: modélisation géométrique à base topologique ; transformations de graphes ; prototypage rapide.

Résumé:

Dans le cadre de la modélisation géométrique, chaque domaine d'application nécessite des logiciels spécifiques, appelés modeleurs. Leurs structures de données sont optimisées en fonction des objets manipulés, et leurs opérations de manipulation d'objets sont dédiées à l'application. Dans le cadre de la modélisation géométrique à base topologique, utilisant les cartes généralisées pour décrire la topologie, les objets géométriques sont représentés par une classe particulière de graphes. La structure topologique (volumes, faces, arêtes, sommets...) des objets est représentée par la structure du graphe, tandis que la géométrie ou toute autre information physique (couleur, forme, position...) sont portées par les nœuds du graphe. Nous avons proposé un langage à base de règles de transformations de graphes pour définir formellement les opérations géométriques.

Notre noyau de modeleur est entièrement paramétrable par l'utilisateur (dimension topologique des objets, nature des plongements, opérations géométriques). Il vérifie la préservation de la cohérence topologique et géométrique des objets. Cette vérification s'effectue statiquement, grâce à des conditions syntaxiques sur les règles définissant les opérations. Les objets sont construits interactivement par application des règles.

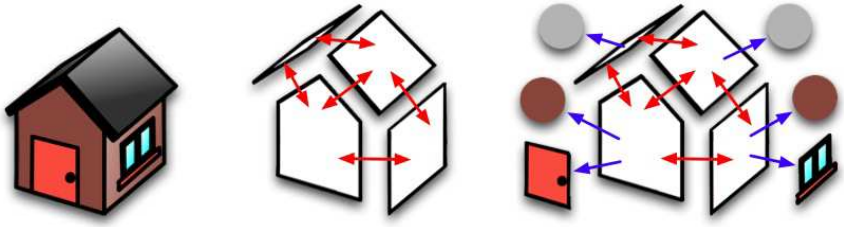
1 Introduction

Traditionnellement, les modèles topologiques et leurs opérations sont définis mathématiquement. C'est le cas notamment des cartes généralisées, encore appelées G-cartes, [Lie91, Lie94] utilisées par JERBOA. La mise en œuvre d'une opération topologique nécessite de prouver que l'objet construit est cohérent vis-à-vis des contraintes liées aux G-cartes, de l'implanter, puis de la tester. Ce processus est long et imparfait. En particulier, seule la topologie est définie mathématiquement, les composantes géométriques n'étant souvent définies qu'au moment de l'implantation.

Les opérations géométriques et topologiques sont parfois définies à l'aide de règles. C'est le cas des L-systèmes [PL91] en dimension 1 et des G-cartes L-systèmes [PTMG08, TGM$^+$09] en dimension 2 et 3. Ces langages sont basés sur un petit nombre d'opérations géométriques qui sont composées pour construire des objets complexes. L'extension de ces langages avec de nouvelles opérations topologiques et de nouveaux plongements est long et complexe.

Nous avons développé un langage à base de règles et les outils associés [PACLG08, Pou09, BPA+10, BAG11]. Le langage est formel car une opération de manipulation d'un objet est décrite à l'aide d'une règle de transformations de graphes au sens de [EEPT06], enrichie de constructions dédiées. De plus, le langage est générique car il permet de définir des opérations géométriques pour toute dimension topologique et tout plongement. Une fois décrites sous forme de règles, les opérations peuvent être appliquées sans aucune implantation : la cohérence des objets construits est garantie par une passe d'analyse statique des règles. Le gain en temps de développement est ainsi très important, au regard de la faible perte de performance en temps d'exécution. Toutes les opérations topologiques classiques ont désormais été définies par des règles [BPA+10].

2 Un langage dédié à la modélisation géométrique



(a) Objet (b) Struct. topologique (c) Struct. topologique plongée

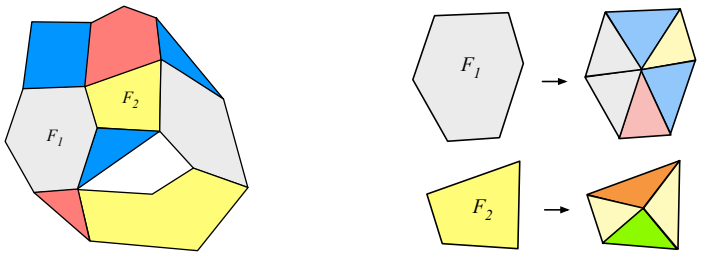
FIGURE 1 – Représentation d'une maison

Les objets modélisés peuvent être de n'importe quelle dimension, et les informations associées aussi variées que la couleur, la masse volumique ou le sens d'ouverture d'une porte. La **modélisation géométrique à base topologique** structure la représentation des objets. Ces derniers sont représentés en premier lieu par leur **topologique**, c'est-à-dire leur décomposition en volumes, faces, arêtes, etc. et les relations de voisinage entre ces derniers. Par exemple, la maison de la figure 1(a) est un volume qui est décomposé en faces selon le graphe de la figure 1(b) sur laquelle les flèches doubles indiquent les liaisons de voisinage. Les informations géométriques et applicatives associées à la structure topologique sont appelées les **plongements**. Dans l'exemple de la maison, nous considérons par exemple les deux plongements suivants : la couleur des faces représentée par des pastilles colorées ; l'inclusion d'une porte ou d'une fenêtre dans les faces. Le graphe ainsi obtenu, où les flèches simples associent les informations de plongement aux faces, est une **structure topologique plongée**. À des fins pédagogiques, l'illustration donnée est un modèle topologique simplifié. Le modèle topo-

logique des G-cartes [Lie91], permet de représenter les faces, les arêtes, les sommets, etc.

En représentant les objets à l'aide de graphes, les opérations de modélisation sont définies par des règles de transformations de graphes [EEPT06], incorporant des variables [Hof05, HJVE06] :

- les **variables topologiques** [PACLG08] permettent de filtrer des cellules topologiques et de les modifier par renommage et suppression des relations de voisinage ;
- les **variables de plongement** [BAG11] permettent de filtrer des plongements et d'en définir de nouveaux à l'aide d'expressions de plongement.



(a) Objet (b) Triangulations

FIGURE 2 – Triangulation de face polyédriques colorées

Le langage obtenu est générique. Il peut être instancié pour toute dimension topologique ou pour tout plongement (défini par ses structures de données et ses opérations de manipulation). Prenons l'exemple de l'opération de triangulation représenté sur la figure 2 dans le cas d'objets polyédriques colorés. Du point de vue topologique, l'opération consiste à subdiviser la face en autant de facettes qu'elle a originellement d'arêtes. Du point de vue géométrique, la transformation consiste à positionner le sommet créé au barycentre de la face et à colorer chaque facette avec la moyenne de la couleur de départ et de la couleur de la face voisine. Par exemple la triangulation de la face F_1, les couleurs des facettes de gauche ne sont pas modifiées, car F_1 n'a pas de voisins à gauche et la facette contiguë à F_2 prend la couleur médiane entre les couleurs de F_1 et F_2.

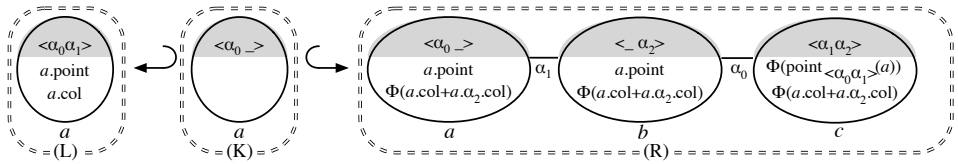


FIGURE 3 – Règle de triangulation

Cette opération est définie par la règle représentée sur la figure 3. La partie gauche L permet de filtrer une face (c'est-à-dire un sous-graphe de type $< \alpha_0\alpha_1 >$ dans une G-carte de dimension 2) dont la couleur est filtrée par la variable de plongement $a.col$. La partie droite R permet de construire les facettes. Par exemple, la suppression des liaisons α_1 du noeuds a (noté par le remplacement de α_1 du noeud a à gauche par un _ à droite) permet de séparer les arêtes de la face afin d'y intercaler les nouvelles arêtes des facettes. De même, le noeud c de type $< \alpha_1\alpha_2 >$ à la même structure que la face filtrée (au renommage près des liaisons de voisinage) ce qui permet de construire un sommet dont le nombre de facettes adjacentes est identique au nombre d'arêtes de la face filtrée. La moyenne Φ est une fonction surchargée du module de plongement, qui permet de calculer dans c le centre de la face à partir des points de plongement des sommets de la face de départ, et de calculer dans a la couleur médiane entre la couleur initiale $a.col$ et la couleur voisine $a.\alpha_2.col$. La partie centrale K permet de raccrocher le motif filtrée (puis le motif réécrit) au reste de l'objet non modifié par la règle.

Des **conditions syntaxiques** sur les règles permettent de garantir la préservation des cohérences topologique et géométrique des objets lors de l'application des règles. De plus, si en général, l'existence de la transformation d'un graphe par une règle ne peut être vérifiée que dynamiquement, nos conditions syntaxiques permettent de garantir l'existence d'une G-carte transformée pour toute G-carte contenant le motif de gauche de la règle.

3 Implantation du noyau de modeleur JERBOA

Le noyau de modeleur JERBOA est programmé en Ocaml [INR]. Il comprend un module Ocaml générique définissant une **structure de G-carte plongée**. L'instanciation de ce module permet de préciser : la dimension topologique de la G-carte, les plongements (c'est-à-dire le type des cellules plongées et le type des données de plongement), et les opérations portant sur les plongements.

JERBOA dispose d'une **fonction de vérification des conditions** sur les règles. En pratique, les règles sont chargées à partir d'un fichier de règles OCaml. Elles sont analysées au chargement.

Le noyau comprend un **moteur d'application de règles**. Lorsque le motif de gauche peut être filtré et les pré-conditions vérifiées, alors la G-carte transformée est calculée. L'efficacité de l'algorithme proposé repose principalement sur une utilisation pertinente de tableaux et d'arbres *union/find*.

Nous avons utilisé le noyau JERBOA pour développer un modeleur polyédrique 3D interactif à l'aide de l'instanciation idoine et d'une interface OpenGL [Gro]. La capture d'écran de la figure 4 montre à droite le fichier des règles, au centre le menu des opérations construit automatiquement à partir des règles chargées, et à gauche l'objet construit. L'utilisateur sélec-

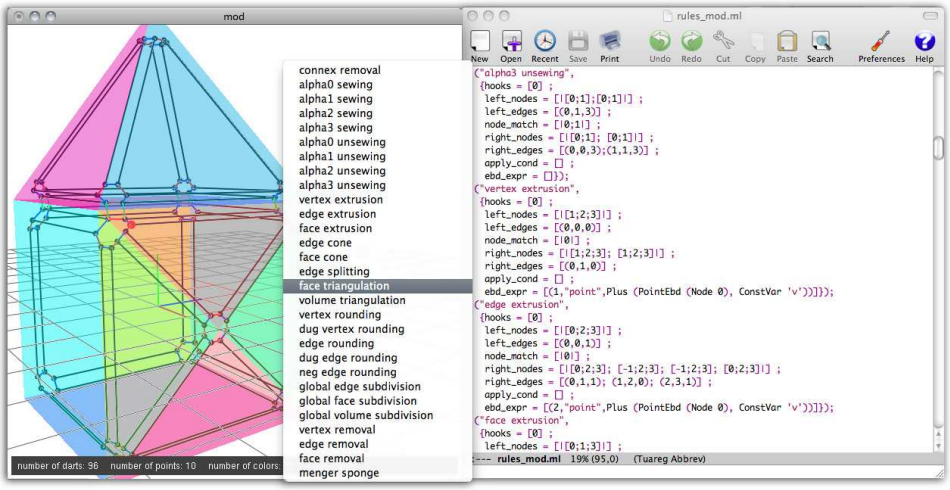


FIGURE 4 – Capture d'écran du modeleur polyédrique 3D

tionne des sommets dans l'objet et l'opération à appliquer en ce sommet. Si l'opération est applicable à l'endroit sélectionné, le nouvel objet est calculé.

4 Conclusion et perspectives

JERBOA permet de prototyper de façon rapide et sûre un modeleur en définissant chaque opération sous la forme de règles. Les conditions syntaxiques sur les règles et leur chargement dynamique sont une aide précieuse pour concevoir rapidement de nouvelles opérations. À titre de comparaison, dans les modeleurs traditionnels, le développement des opérations fait souvent appel à des fonctions auxiliaires dont l'objectif affiché est de rétablir la cohérence des objets après l'application d'une opération. L'évaluation de la performance sur la version limitée aux transformations topologiques [BPA+10] a montré que la complexité des opérations implantées sous forme de règles était du même ordre que leurs versions développées manuellement.

La prochaine étape de ce travail consiste d'une part à valider l'approche sur des domaines d'applications comme la géologie ou l'architecture par exemple. D'autre part, des extensions sont encore à apporter au langage de règles, pour permettre par exemple, la composition de règles pour construire un traitement complexe. Enfin, JERBOA et le langage sous-jacent est spécialisé pour le modèle topologique des cartes généralisées. Il serait intéressant d'utiliser la même approche pour d'autres modèles topologiques pour utiliser par exemple des modèles plus légers en mémoire (comme les cartes combinatoires) ou plus puissants (comme les modèles hiérarchiques).

Références

[BAG11] T. Bellet, A. Arnould, and P. Le Gall. Rule-based transformations for geometric modelling. In *Proceedings 6th International Workshop on Computing with Terms and Graphs, TERM-GRAPH*, volume 48 of *EPTCS*, pages 20–37, 2011.

[BPA+10] T. Bellet, M. Poudret, A. Arnould, L. Fuchs, and P. Le Gall. Designing a topological modeler kernel : a rule-based approach. In *Shape Modeling International (SMI'10) Shape Modeling International (SMI'10)*, Aix-en-Provence, France, 2010.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation (Monographs in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006.

[Gro] Khronos Group. The Open Graphics Library (OpenGL). http ://www.opengl.org.

[HJVE06] B. Hoffmann, D. Janssens, and N. Van Eetvelde. Cloning and expanding graph transformation rules for refactoring. *Electronic Notes in Theoretical Computer Science*, 152 :53–67, 2006.

[Hof05] B. Hoffmann. Graph transformation with variables. *Formal Methods in Software and System Modeling*, 3393 :101–115, 2005.

[INR] INRIA. The OCaml Language. http ://www.ocaml.org.

[Lie91] P. Lienhardt. Topological models for boundary representation : a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1) :59–82, jan 1991.

[Lie94] P. Lienhardt. N-dimensional generalised combinatorial maps and cellular quasimanifolds. *International Journal on Computational Geometry and Applications (IJCGA)*, (3) :275–324, 1994.

[PACLG08] M. Poudret, A. Arnould, J.-P. Comet, and P. Le Gall. Graph transformation for topology modelling. In *4th International Conference on Graph Transformation (ICGT'08)*, volume 5214 of *LNCS*, pages 147–161, Leicester, United Kingdom, September 2008. Springer.

[PL91] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants (the virtual laboratory). 1991.

[Pou09] M. Poudret. *Transformations de graphes pour les opérations topologiques en modélisation géométrique, Application à l'étude de la dynamique de l'appareil de Golgi*. Thèse, Université d'Évry val d'Essonne, Programme Epigénomique, October 2009.

[PTMG08] A. Peyrat, O. Terraz, S. Merillou, and E. Galin. Generating vast varieties of realistic leaves with parametric 2gmap L-systems. *The visual Computer*, 24(7) :807–816, 2008.

[TGM+09] O. Terraz, G. Guimberteau, S. Mérillou, D. Plemenos, and D. Ghazanfarpour. 3Gmap L-systems : an application to the modelling of wood. *The Visual Computer*, 25(2) :165–180, 2009.